Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA ILLINOIS 61801

CAC Document No. 12

THE QR-ALGORITHM

by

Masako Ogura

September 1, 1971

CAC Document No. 12

THE QR-ALGORITHM

by

Masako Ogura

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

September 1, 1971

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the U.S. Army Research Office - Durham under Contract No. DAHC04-72-C-0001.

ABSTRACT

The implementation of QR-algorithm on ILLIAC IV is described. An ASK subroutine for computing all eigenvalues of a real Hessenberg matrix of order less than or equal to 64 by this algorithm is attached. The QR-transformation consists of the decomposition of the matrix A_k into the product of a unitary matrix Q_k and an upper triangular matrix R_k, and forming A_{k+1} by post-multiplying R_k by Q_k, where $A_1 = A$ is the original matrix. All eigenvalues are either isolated on the diagonal or are eigenvalues of a 2 x 2 diagonal submatrix as $k \to \infty$.

TABLE OF CONTENTS

1. INTRODUCTION

An ASK program for finding all eigenvalues of a real Hessenberg matrix of order less than or equal to 64 is written and tested on the B5500 simulator. The QR-algorithm for real Hessenberg matrices described herein is that of Martin et al [2]. The ILLIAC IV computer time required for performing one iteration (computation of A_{k+2} from A_k, refer to 4.1) on a 64 x 64 matrix is approximately 10 millisecond.

The necessary information for using this program is given in Section 2. The test result of this program on a 4 x 4 real Hessenberg matrix is given in Section 3. In Section 4, the outline of the QR-algorithm is given and Section 5 is devoted to the actual programming technique to implement this algorithm on the ILLIAC IV computer. The flow chart and ASK program are attached as Appendices 1 and 2 respectively.

2. USAGE

This program assumes that the given real Hessenberg matrix is stored in the core memory in the straight storage scheme so that each row is stored across the PE's, starting with PE0. The real and imaginary parts of the eigenvalues found are to be stored in the two rows in the PE memory specified by the user. The original matrix is destroyed and replaced by the matrix which results from the QR transformations. The content of ACAR2 and ACAR3 are destroyed since the ACAR3 is used for linkage between the subroutine and main program and ACAR2 is used for passing the address of parameters to the subroutine.

2.1 Calling sequence

Calling sequence for this subroutine is:

CALL HQR (N, A, WR, WI, IT)

where A designates the first row of the matrix and is declared in the main program as

$$
\begin{array}{lllll}
\text{A:} & \text{DATA} & a_{00}, a_{01}, \cdots\cdots, & a_{0,N-1}, & (0.0)M, \\
 & & a_{10}, a_{11}, \cdots\cdots, & a_{1,N-1}, & (0.0)M, \\
 & & \cdots\cdot & & \\
 & & \cdots\cdot & & \\
 & & a_{N-1,0}, a_{N-1,1}, \cdots, & a_{N-1,N-1}, & (0.0)M;
\end{array}
$$

where M = 64 - N.

N is the size of the matrix declared as

N: EQU 64;

or

DEFINE N = 64 ##;

or given as an integer, i.e., 64.

WR and WI are the rows in the PE memory where resulting real and imaginary parts of the eigenvalues respectively are to be stored. They are declared as

WR: BLK 1; and WI: BLK 1;

IT is the row vector in memory to which the number of iterations required for finding each eigenvalue is to be placed. This is declared as

IT: BLK 1;

If zero is placed in place of IT, it is to be considered that a user does not want to know the number of iterations required. The CALL macro should be defined in the user's program as:

```
DEFINE CALL  &NAME (&PARAMETERS) =
        &IF  &SIGN (&MFIELD(&NAME))
             &THEN EXTERNAL &NAME; &FI
        &IF  &EMPTY (&PARAMETERS) &THEN &ELSE
        BEGIN BLOCK
             BEGIN USE (63)
             LIST:  DATA &PARAMETERS
             END;
             CLC(2);
             SLIT(2) LIST;
        END; &FI
        CLC(3);
        SLIT(3) &NAME;
        EXCHL(3) &ICR ##;
```

2.2 Core storage used

This routine uses 500 words of PE memory for storing instructions. One row is used for storing PE numbers and three additional rows are used for temporary storage. ADBO ~ 32 are also used.

2.3 Constant

EPS (ϵ), the constant which is used to test the convergence (4.1), is taken as 10^{-10} in this program. If a user wants to change the value of this constant, he may insert EPS: DATA (desirable value); in place of

EPS: DATA @ - 10; .

3. EXAMPLE

A test of this program was made on the B5500 simulator for the matrix:

$$\begin{bmatrix} 5.0 & -2.0 & -5.0 & -1.0 \\ 1.0 & 0.0 & -3.0 & 2.0 \\ 0.0 & 2.0 & 2.0 & -3.0 \\ 0.0 & 0.0 & 1.0 & -2.0 \end{bmatrix}$$

with $\epsilon = 10^{-8}$. The comparison of the eigenvalues obtained by this program to the exact values is given in Table 1. The selection of this small matrix and a relatively large ϵ was made because of the speed of the SSK simulator on the B5500; the execution speed ratio of the simulator to the ILLIAC IV is approximately $1:10^6$.

Table 1

Eignevalues obtained on B5500 simulator	Exact eigenvalues
3.999999997867	4.0
1.000000001066 + 1.999999999928i	1.0 + 2.0i
1.000000001066 - 1.999999999928i	1.0 - 2.0i
-1.000000000000	-1.0

4. QR-ALGORITHM

4.1 Brief outline of the QR-algorithm

The QR-transformation consists of the decomposition of the matrix A_k into the product of a unitary matrix Q_k and an upper triangular matrix R_k, and forming A_{k+1} by post-multiplying R_k by Q_k. Thus

$$A_{k+1} = R_k Q_k \qquad \text{where} \qquad A_k = Q_k R_k, \tag{1}$$

therefore

$$A_{k+1} = Q_k^* A_k Q_k$$

where $A_1 = A$ is the original matrix. It can be shown in general that A_k tends to a form in which $a^{(k)}_{i+1, i}\ a^{(k)}_{i+2, i+1} = 0$ for $i = 0, 1, \ldots, N-3$ as k increases. All eigenvalues are therefore either isolated on the diagonal or they are eigenvalues of a 2 x 2 diagonal submatrix. The amount of calculations involved in a QR step is greatly reduced if the matrix A is in the Hessenberg (or almost triangular) form. Since there are several stable methods available to reduce a general matrix to this form (ASK program HSBG is written for this purpose), the QR-algorithm is used after such reduction.

In order to achieve rapid convergence, it is essential that the origin shifts be applied and that each shift be close to an eigenvalue of the matrix. The QR-algorithm with shift of an origin s_k is expressed as:

$$A_{k+1} = R_k Q_k + s_k I \qquad \text{where} \qquad A_k - s_k I = Q_k R_k \tag{2}$$

or in other words

$$A_{k+1} = Q_k^* A_k Q_k.$$

However, even when A_1 is real, some of the eigenvalues may be complex. If the transformation (2) is carried out with a complex value of s_k, A_{k+1} is in general a complex matrix. This deficiency can be overcome by performing two steps of (2) with shifts of s_k and s_{k+1} respectively. Since s_k and s_{k+1} are both real or complex conjugate in this transformation, A_{k+2} should be always real. This transformation is described as

$$A_{k+2} = Q^*_{k+1} Q^*_k A_k Q_k Q_{k+1}$$

and $$(Q_k Q_{k+1})(R_{k+1} R_k) = (A_k - s_k I)(A_k - s_{k+1} I) \qquad (3)$$

One method of calculating A_{k+2} by (3) would be to form the real matrix $\Gamma = (A_k - s_k I)(A_k - s_{k+1} I)$, computing its unitary-triangular decompositon to obtain $Q_k Q_{k+1}$ and transform A_k by means of this, thus giving A_{k+2}. This process requires a prohibitive amount of work, but it is shown [1] that when the matrix is in the Hessenberg form, it is unnecessary to compute more than the first column of Γ, and that this immediately gives the transformation to be applied to A_k.

4.2 Practical computation

If (3) is rewritten as

$$A_{k+2} = W^* A_k W \qquad \text{and} \qquad W^* \Gamma = \Delta$$

where $W = Q_k Q_{k+1}$ and Δ is the triangular matrix $R_{k+1} R_k$, W^* is a unitary matrix which reduced Γ to the triangular Δ, and W is composed of N unitary factors of the form $M_i = \begin{bmatrix} I_{i-1} & 0 \\ 0 & B_i \end{bmatrix}$ so that $W = M_1 M_2 \dots M_k$. From the form of each M_i we see that the first column of W is equal to the first column of M_1, and this is any unitary matrix, the transpose of which eliminates the elements of the first column of Γ below diagonal.

Since we wish to transform A_k to A_{k+2} by W, we first operate on A_k with M_1. This will change the first three rows and columns of A_k since the first column of Γ contains only three non-zero elements. It follows;

$$A_k = \begin{bmatrix} a_{00} & a_{01} & a_{02} \cdots\cdots & a_{0,N-1} \\ a_{10} & a_{11} & a_{12} \cdots\cdots & a_{1,N-1} \\ & a_{21} & a_{22} \cdots\cdots & a_{2,N-1} \\ & & & \\ & & & \\ & & & a_{N-1,\,N-1} \end{bmatrix}$$

$$\to \begin{bmatrix} \underline{a'_{00}} & \underline{a'_{01}} & \underline{a'_{02}} & \underline{a_{03}} \cdots\cdots & \underline{a_{0,N-1}} & \\ \underline{a'_{10}} & \underline{a'_{11}} & \underline{a'_{12}} & \underline{a_{13}} \cdots\cdots & \underline{a_{1,N-1}} & \\ \underline{a'_{20}} & \underline{a'_{21}} & \underline{a'_{22}} & \underline{a_{23}} \cdots\cdots & \underline{a_{2,N-1}} & \\ a'_{30} & a'_{31} & a'_{32} & a_{33} \cdots\cdots & a_{3,N-1} & \\ & & & a_{43} \cdots\cdots & a_{4,N-1} & \\ & & & \cdots\cdots\cdots & & \\ & & & \cdots\cdots\cdots & & \\ & & & & & a_{N-1,\,N-1} \end{bmatrix}$$

where the elements changed by the row and column operations are underlined and primed respectively. The resulting matrix is no longer in the Hessenberg form and since A_{k+2} is in the Hessenberg form, we can say that the matrices $M_2 \cdot M_3 \cdot \ldots M_N$ reduce $M_1^* A_k M_1$ to the Hessenberg form. In the practical computation, therefore, it is necessary to compute only the first column of Γ.

After each iteration (we call the calculation of A_{k+2} from A_k an iteration), all subdiagonal elements of A_{k+2} are examined to see if any of them are "negligibly small." If so, the eigenproblem for the current matrix splits into that for two or more Hessenberg matrices of smaller sizes, and the iterations continue with the submatrix in the bottom right-hand corner. It may happen that while no individual subdiagonal element is sufficiently small to be regarded as negligible, the product of two consecutive elements may be small enough to permit us to work with a submatrix. Therefore the examination of the matrix A_{k+2} is performed to see if any two consecutive subdiagonal elements are small.

5. PROGRAM DESCRIPTION

5.1 Search for negligible subdiagonal elements

We assume that the size of the matrix under consideration is $(n + 1) \times (n + 1)$ where n takes integer values between 1 and N - 1. If the last negligible subdiagonal element is in position $(\ell, \ell - 1)$, it is required only to work on the submatrix in the rows and columns ℓ to n. If none of the subdiagonal elements are negligible, ℓ is taken to be 0. The following criterion is used

$$|a_{\ell, \ell - 1}| \leq \epsilon (|a_{\ell - 1, \ell - 1}| + |a_{\ell, \ell}|).$$

This criterian examines whether $a_{\ell, \ell - 1}$ is negligible compared to the local diagonal elements.

On each PE ℓ, the following computations are simultaneously performed:

$$f(\ell) \equiv |a_{\ell, \ell - 1}| - \epsilon (|a_{\ell - 1, \ell - 1}| + |a_{\ell, \ell}|$$
$$\text{for } \ell = 1, 2, \ldots, n.$$

If $f(\ell)$ is negative, 1 is placed in ℓth bit of the ACAR. Then searching is made for the lowest bit of the ACAR which contains 1. For example, in the following case:

	0	1	2	3 . . .	19	20	. . . .	63
ACAR	0	0	1	0 . . .	0	1	0 0 . . . 0	0

ℓ is set to be 20.

Then test is made if $\ell = n$ or $\ell = n - 1$. If $\ell = n$, one eigenvalue is found in the place (n, n) and the matrix is deflated by 1, and n is decreased by one. In the case that $\ell = n - 1$, two eigenvalues are found as the eigenvalues of the bottom-right hand corner 2×2 submatrix. Then two columns and rows are deflated and n is decreased by 2.

5.2 Shifts of origin

The shifts of origin at each stage are taken to be the two roots, s_k and s_{k+1}, of the 2 x 2 matrix in the bottom right-hand corner of the current A_k. This gives

$$s_k + s_{k+1} = a_{n-1, n-1} + a_{n, n}$$

and

$$s_k s_{k+1} = a_{n-1, n-1}\, a_{n,n} - a_{n-1, n}\, a_{n, n-1}\ . \qquad (4)$$

In some rare cases, the process fails to converge with these shifts of origin. An example of such failure is provided by matrices of the type:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ & & 1 & 0 & 0 \\ & & & 1 & 0 \end{bmatrix} .$$

Here the shifts of origin, given in (4), are both zero, and since the matrix is orthogonal, it is invariant with respect to the QR transformation without shifts. However, if one iteration is performed with any shifts of origin which are loosely related to the norm of the matrix, the convergence is very rapid. Therefore in the case where ten iterations do not produce an eigenvalue, the usual shifts s_k and s_{k+1} are replaced by shifts defined by

$$s_k + s_{k+1} = 1.5\ (|a_{n, n-1}| + |a_{n-1, n-2}|)\ .$$

$$s_k s_{k+1} = (|a_{n, n-1}| + |a_{n-1, n-2}|)^2 \qquad (5)$$

This strategy is used again after 20 unsuccessful iterations. If 30 unsuccessful iterations are needed then a failure indication is given.

In this program, ITS is the name of an ADB where the iteration count is stored. When ITS $\neq$ 10, 20, we form $S = s_k + s_{k+1}$ and $Y = s_k s_{k+1}$ on a PE n according to (4) and store in ADB's. When ITS = 10 or 20, scheme (5) is used for computing S and Y. If ITS = 30, it is assumed that this algorithm fails to produce eigenvalues. As a result, only eigenvalues computed prior to this point are given in WR and WI.

5.3 Search for two consecutive small subdiagonal elements

After determining ℓ, (5.1), the submatrix in the rows ℓ to n are examined to see if any two consecutive subdiagonal elements are small enough to work with an even smaller submatrix. To test if we are to start at the row m, we compute the elements p_m, q_m and r_m such that

$$
\begin{aligned}
p_m &= a_{mm}^2 - a_{mm}(s_k + s_{k+1}) + s_k s_{k+1} + a_{m,m+1} a_{m+1,m} \\
q_m &= a_{m+1,m}(a_{mm} + a_{m+1,m+1} - s_k - s_{k+1}) \\
r_m &= a_{m+2,m+1} a_{m+1,m} .
\end{aligned}
\tag{6}
$$

The criterion applied is

$$
\begin{aligned}
&|a_{m,m-1}| (|q_m| + |r_m|) \\
&\qquad \leq \epsilon |p_m| (|a_{m+1,m+1}| + |a_{m,m}| + |a_{m-1,m-1}|)
\end{aligned}
\tag{7}
$$

where we test whether or not the elements which appear in the positions $(m + 1, m)$, $(m + 2, m + 1)$ are negligible compared with the three local diagonal elements $a_{m+1,m+1}$, $a_{m,m}$ and $a_{m-1,m-1}$.

Here we take m to be the largest integer ($\geq \ell$) for which condition (6) is satisfied.

For this computation, the mode bits are turned on for PE ℓ through PE n. The p_m, q_m and r_m for $m = \ell, \ell - 1, \ldots, n$ are computed according to (6) on all PE's whose mode bits are turned on, and comparison is made to see whether (7) is satisfied. If (7) is satisfied on PE **m**, 1 is placed in the mth bit of the ACAR. Then the search is made for the lowest bit of the ACAR which contains 1, and m is set equal to this bit number. If no 1 is found in the ACAR, m is taken to be ℓ.

5.4 Double QR-transformation

A_{k+2} is computed by applying the QR-double transformation to A_k in such a way that $A_{k+2} = N_n^* \ldots N_m^* A_k N_m \ldots N_n$

where $N_i^* = I - \dfrac{U_i U_i^*}{2K_i^2}$ and $U_i = (p_i \pm t_i, q_i, r_i, 0 \ldots 0)$.

Here

$$p_i = a_{ii}^2 - a_{ii}(s_k + s_{k+1}) + s_k s_{k+1} + a_{i,i+1} a_{i+1,i}$$

$$q_i = a_{i+1,i}(a_{ii} + a_{i+1,i+1} - s_k - s_{k+1})$$

$$r_i = a_{i+2,i+1} a_{i+1,i} \qquad \text{for } i = m$$

and

$$p_i = a_{i,i-1} \pm t_i, \qquad q_i = a_{i+1,\, i-1}$$

$$\text{and } r_i = a_{i+2,i-1} \qquad \text{for } i \neq m.$$

t_i and $2K_i^2$ are defined as

$$t_i = \pm\sqrt{p_i^2 + q_i^2 + r_i^2}$$

$$2K_i^2 = t_i^2 \mp p_i t_i.$$

Row modification:

For $i = m, m+1, \ldots, n$, the elements of $N_i^* A_k^{(i)} \equiv N_i^* (N_{i-1}^* \ldots N_m^* A_k N_m \ldots N_{i-1})$ are different from those of $A_k^{(i)}$ in only three rows, i.e., ith, (i + 1)th and (i + 2)th rows. These new elements are computed in the following way with the elements of $A_k^{(i)}$ denoted by a_{hj}:

$$(i,j) - \text{element} = a_{ij} - [(p_i \pm t_i)a_{ij} + q_i a_{i+1,j} + r_i a_{i+2,j}]\frac{1}{t_i}$$

$$(i+1,j) - \text{element} = a_{i+1,j} - [(p_i \pm t_i)a_{ij} + q_i a_{i+1,j} + r_i a_{i+2,j}]\frac{q_i}{2K_i^2}$$

$$(i+2,j) - \text{element} = a_{i+2,j} - [(p_i \pm t_i)a_{ij} + q_i a_{i+1,j} + r_i a_{i+2,j}]\frac{r_i}{2K_i^2}$$

$$\text{for } j = i, i+1, \ldots, n$$

In the actual computation, t_i and $2K_i^2$ are first computed and then p_i, q_i and r_i are found and stored in ADB's. The mode bits of PE's which contain $a_{i,i}$, $a_{i,i+1}$, ... , $a_{i,n}$ are turned on and $c_i \equiv (p_i \pm t_i)a_{ij} + q_i a_{i+1,j} + r_i a_{i+2,j}$ are computed on these PE's. The computation of new elements

$$(i,j) \text{ - element } = a_{ij} - \frac{c_i}{t_i}$$

$$(i,j+1) \text{ - element } = a_{i,j+1} - \frac{c_i q_i}{2K_i^2}$$

$$(i,j+2) \text{ - element } = a_{i,j+2} - \frac{c_i r_i}{2K_i^2}$$

are then performed.

Column Modification:

Similarly $(N_i^* A_k^{(i)})N_i$ is computed from $N_i^* A_k^{(i)}$ for $i = m, m + 1,$ . . . , n in the following way where the element of matrix $N_i^* A_i^{(k)}$ are denoted as $a_{j,h}$:

$$(j,i) \text{ - element } = a_{ji} - [(p_i \pm t_i)a_{ji} + q_i a_{j,i+1} + r_i a_{j,i+2}] \frac{1}{t_i}$$

$$(j,i+1) \text{ - element } = a_{j,i+1} - [(p_i \pm t_i)a_{ji} + q_i a_{j,i+1} + r_i a_{j,i+2}] \frac{q_i}{2K_i^2}$$

$$(j,i+2) \text{ - element } = a_{j,i+2} - [(p_i \pm t_i)a_{ji} + q_i a_{j,i+1} + r_i a_{j,i+2}] \frac{r_i}{2K_i^2}$$

$$\text{for } j = \ell, \ldots\ldots, \min [i+3,n].$$

As in the row modification, $c_i' = (p_i \pm t_i)a_{ji} + q_i a_{j,i+1} + r_i a_{j,i+2}$ are computed on the PE's which contain a_{ji} for j from ℓ through min [i + 3, n], then the computations of new (j,i) - element, (j, i + 1) - element and (j, i + 2) - element are performed on the corresponding PE's.

5.5 Computation of eigenvalues

The eigenvalues are calculated as the last step of program after $a_{i+1,i}$ or $a_{i,i-1}$ $a_{i+1,i}$ become negligibly small for all $0<i<n-2$. At each

time when the matrix is deflated by 2, 1 is placed in position n of ADB named SOLV. If the transformation is carried out successfully, SOLV looks like the following,

$$\begin{array}{cc} & \begin{array}{ccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \cdots\cdots & 62 & 63 \end{array} \\ \text{SOLV} & \boxed{\begin{array}{ccccccccccccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \cdots\cdots & 0 & 1 \end{array}} \end{array}.$$

Here, the eigenvalues we are looking for are a_{00}, a_{33}, a_{44}, a_{55}, a_{88}, $a_{11,11}$, ... , $a_{61,61}$ and eigenvalues of the following 2 x 2 matrices:

$$\begin{bmatrix} a_{11} & \\ & a_{22} \end{bmatrix}, \begin{bmatrix} a_{66} & \\ & a_{77} \end{bmatrix}, \begin{bmatrix} a_{99} & \\ & a_{10,10} \end{bmatrix} \text{ and } \begin{bmatrix} a_{62,62} & \\ & a_{63,63} \end{bmatrix}.$$

These eigenvalues can be calculated simultaneously on the corresponding PE's.

5.6 The number of iteration required

After each deflation, the number of iterations required to find each eigenvalue (or two eigenvalues) appears in ADB named ITS. This is placed as the nth element of the row vector specified by the user as the fifth parameter in the calling sequence. If this parameter is specified as 0, the number of iterations is not stored anywhere.

REFERENCES

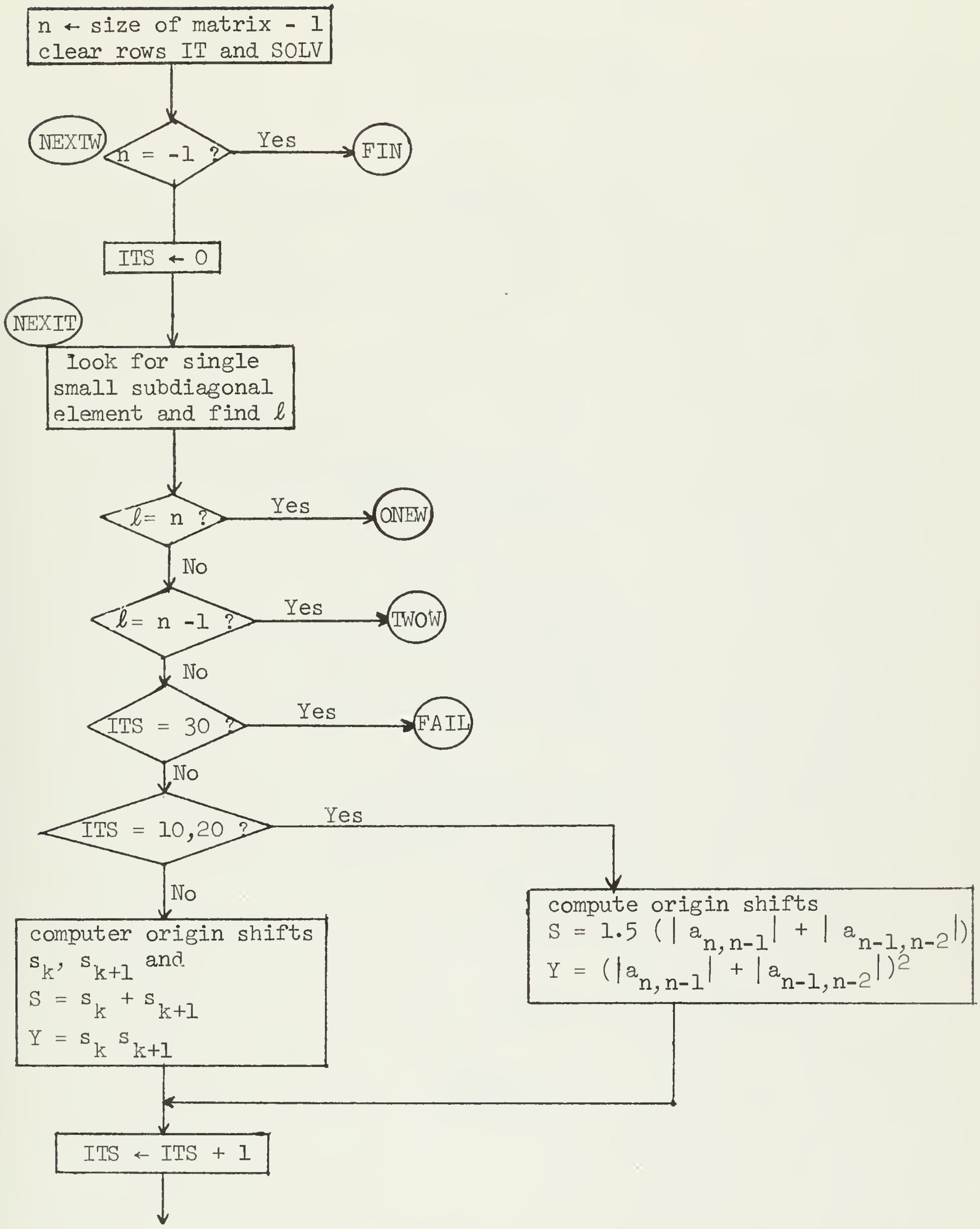
1. J.G.F. Frances: "The QR Transformation--A Unitary Analogue to the LR Transformation," Parts I and II, Comp. Journal, 4, 265-271 and 332-345 (1961/62).

2. R.S. Martin, G. Peters and J.H. Wilkinson: The QR Algorithm for Real Hessenberg Matrices. Numer. Math., 14, 219-231 (1970).

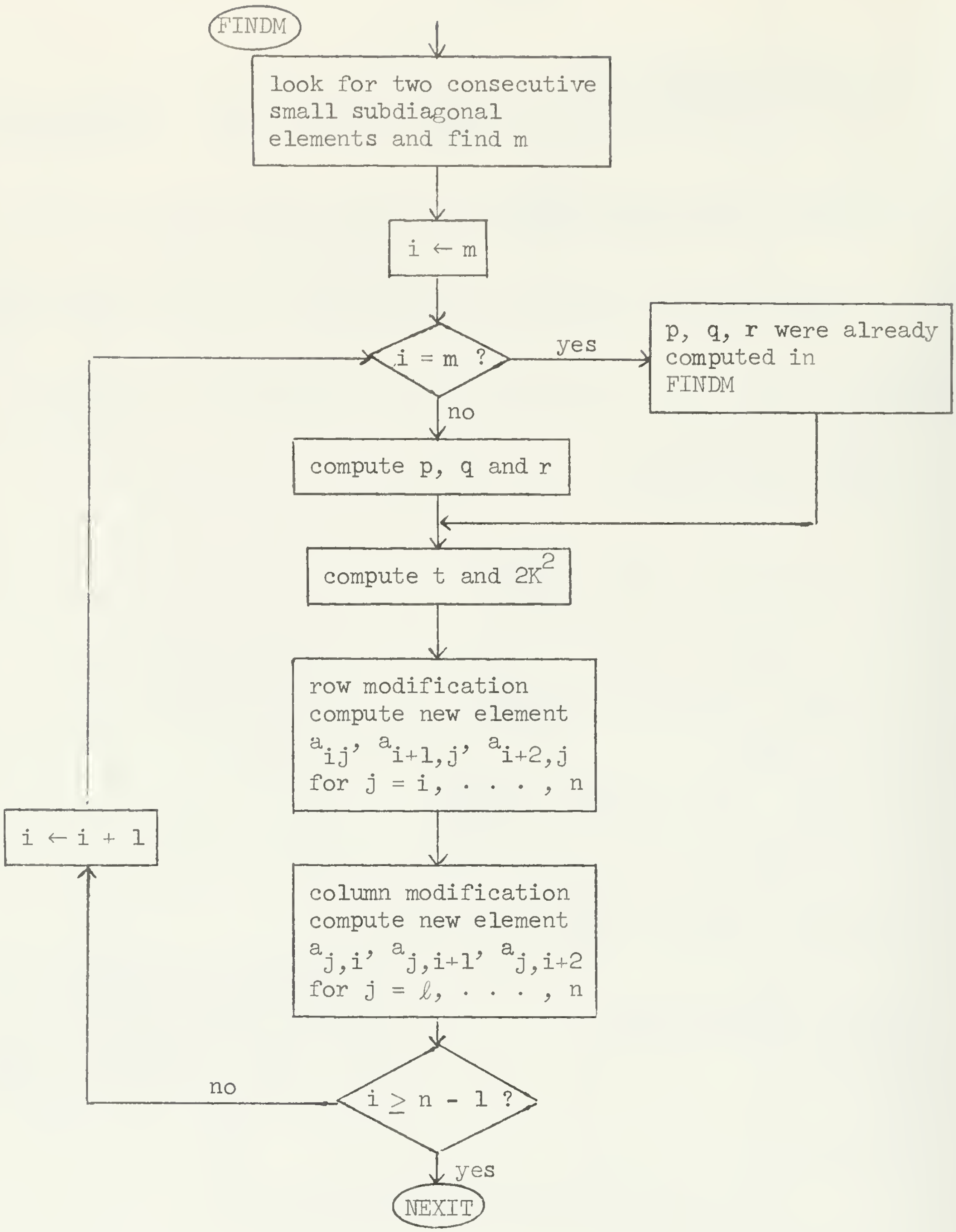
3. B.N. Parlett: The LU and QR Algorithm. Mathematical Methods for Digital Computers, Vol. 2, A. Ralston and H. Wilf, Editors, J. Wiley, 1968.

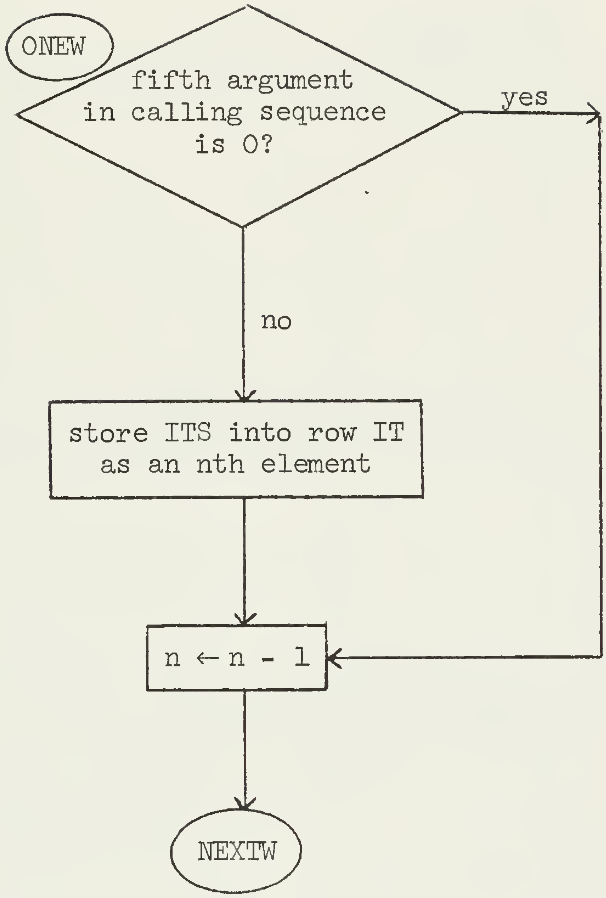
4. J.H. Wilkinson: The Algebraic Eigenvalue Problem. Oxford University Press (1965).

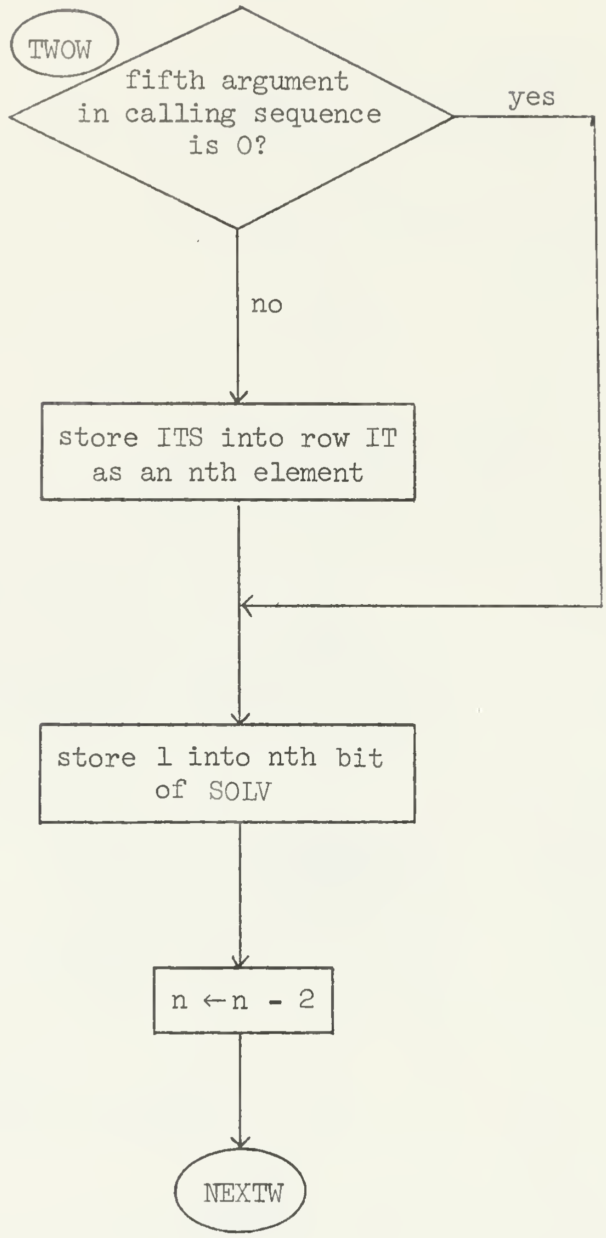
APPENDIX 1

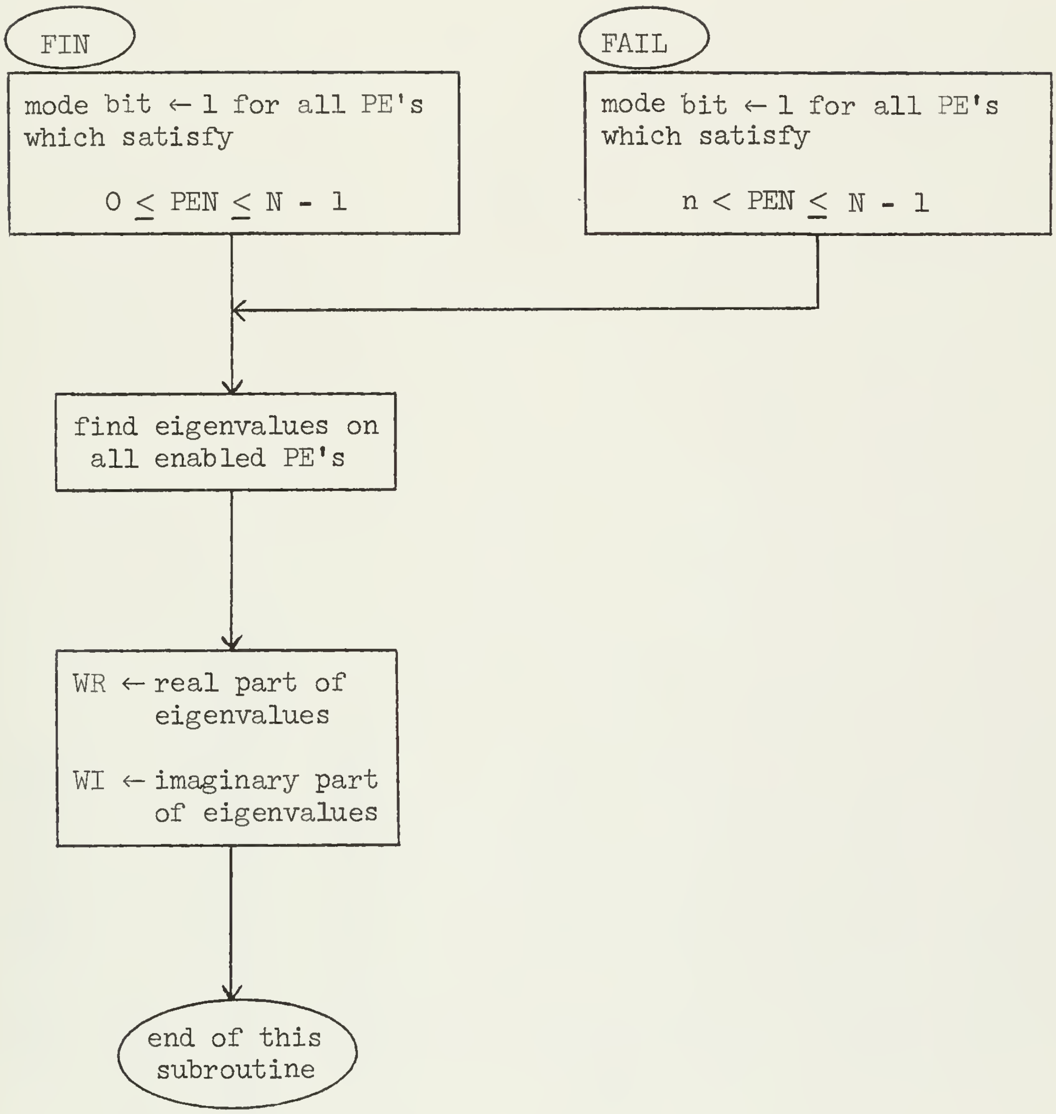


n ← size of matrix - 1
clear rows IT and SOLV
NEXTW
n = -1 ?
Yes
FIN
ITS ← 0
NEXIT
look for single small subdiagonal element and find ℓ
ℓ = n ?
Yes
ONEW
No
ℓ = n -1 ?
Yes
TWOW
No
ITS = 30 ?
Yes
FAIL
No
ITS = 10,20 ?
Yes
No
computer origin shifts
s_k, s_{k+1} and
$S = s_k + s_{k+1}$
$Y = s_k\ s_{k+1}$
compute origin shifts
$S = 1.5\ (|a_{n,n-1}| + |a_{n-1,n-2}|)$
$Y = (|a_{n,n-1}| + |a_{n-1,n-2}|)^2$
ITS ← ITS + 1


FINDM
look for two consecutive small subdiagonal elements and find m
i ← m
i = m ?
yes
no
p, q, r were already computed in FINDM
compute p, q and r
compute t and 2K^2
row modification compute new element a_{ij}, $a_{i+1,j}$, $a_{i+2,j}$ for j = i, . . . , n
column modification compute new element $a_{j,i}$, $a_{j,i+1}$, $a_{j,i+2}$ for j = ℓ, . . . , n
i ← i + 1
i ≥ n - 1 ?
no
yes
NEXIT


ONEW
fifth argument in calling sequence is 0?
yes
no
store ITS into row IT as an nth element
n ← n - 1
NEXTW


TWOW
fifth argument in calling sequence is 0?
yes
no
store ITS into row IT as an nth element
store 1 into nth bit of SOLV
n ←n - 2
NEXTW


FIN
mode bit ← 1 for all PE's which satisfy
0 ≤ PEN ≤ N - 1
FAIL
mode bit ← 1 for all PE's which satisfy
n < PEN ≤ N - 1
find eigenvalues on all enabled PE's
WR ← real part of eigenvalues
WI ← imaginary part of eigenvalues
end of this subroutine

```
        BEGIN
.HQSVO:EQU          $D0;
.HQSV1:EQU          $D1;
.HQSV3:EQU          $D2;
.HQSVE:EQU          $D3;
.N    : EQU         $D4;
.H    : EQU         $D5;
.WR   : EQU         $D6;
.WI   : EQU         $D7;
.IT   : EQU         $D8;
.MASK1:EQU          $D9;
.ONE  : EQU         $D10;
.ITS  : EQU         $D11;
.L    : EQU         $D12;
.M    : EQU         $D13;
.SOLV: EQU          $D14;
.NOTLAST:EQU        $D15;
.S    : EQU         $D16;
.Y    : EQU         $D17;
.P    : EQU         $D18;
.Q    : EQU         $D19;
.R    : EQU         $D20;
.XX   : EQU         $D21;
.X    : EQU         $D22;
.Z    : EQU         $D23;
.JBIT: EQU          $D24;
.J    : EQU         $D25;
.SAVI: EQU          $D26;
.SAVJ: EQU          $D27;
.SVO:  EQU          $D28;
.SV1:  EQU          $D29;
%
TMP1 : BLK          1;
TMP2 : BLK          1;
TMP3 : BLK          1;
P    : EQU          TMP1;
Q    : EQU          TMP2;
R    : EQU          TMP3;
       EXTERNAL     SQRT;
       FILL         128;
PEN  : DATA         0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21
                    ,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,
                    40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,
                    58,59,60,61,62,63;
EPS  : DATA         1.0P-10;
%  MATRIX IS ASSUMED TO BE IN STRAIGHT STORAGE
%   STORE PARAMETERS OF SUBROUTINE INTO ADBS
HQR[ENTRY]::
       STL(3)       .HQSV3;               % SAVE CONTENT OF ACAR3
       STL(1)       .HQSV1;               % SAVE CONTENT OF ACAR1
       STL(0)       .HQSVO;               % SAVE CONTENT OF ACARO
       SETC(3)      E;
       STL(3)       .HQSVE;               % SAVE E BIT
       LOAD(2)      $C0;
       LIT(1)       =1;
```

```
        STL(1)     .ONE;               % .ONE:=1
        CSUB(0)    $C1;
        STL(0)     .N;                 %.N:=(SIZE OF MATRIX)-1
        ALIT(2)    =1;
        LOAD(2)    $C0;
        CSHR(0)    6;
        STL(0)     .H;                 %H:=PE ADDRESS OF H
        ALIT(2)    =1;
        LOAD(2)    $C0;
        CSHR(0)    6;
        STL(0)     .WR;                %.WR:=PE ADDRESS OF WR
        ALIT(2)    =1;
        LOAD(2)    $C0;
        CSHR(0)    6;
        STL(0)     .WI;                %.WI:=PE ADDRESS OF WI
        ALIT(2)    =1;
        LOAD(2)    $C0;
        CSHR(0)    6;
        STL(0)     .IT;                %.IT:=PE ADDRESS OF IT
%       INITIALIZATION
        LIT(3)     =0000000000FFFFFF:16;
        STL(3)     .MASK1;
        CLC(1)     ;
        STL(1)     .SOLV;              % CLEAR CONTENT OF .SOLV
        LDL(0)     .N;                 %ACAR0:=(SIZE OF MATRIX)-1
%  BEGINNING OF N-LOOP, ACARO CONTAINS N
NEXTWI:LIT(3)      0.0,-1;
        CSUB(3)    $C0;                % ACAR3:=-1-N
        SETE       E OR -E;
        SETE1      E OR E;
        ZERF(3)    ,NEXW1;
        JUMP       FIN;                % IF N=-1 GO TO FIN
NEXW1: LIT(2)      =1,30,0;
        STL(2)     .ITS;         %ITERATION COUNT .ITS IS SET TO 0
% DOUBLE QR ITERATION LOOP
%  LOOK FOR SINGLE SMALL DIAGONAL ELEMENT
NEXTIT:  LDX       PFN;          % LOAD PE NUMBER TO RGX
        IXL        =0(0);
        JXE        =0(0);
        SETE       I AND E;
        SETH       J OR E;             %H:=1 IF RGX LEQ N
        SETE       E OR -E;
        SETE       H AND E;
        SETE1      E OR E;             %E:=1 IF RGX LEQ N
        LDL(3)     .H;
        LDA        *0(3);              %SA:=H[L,L]
        SAP        ;                   % TAKE ABSOLUTE VALUE
        RTL        $A,1;               %SR:=H[L-1,L-1]
        ADRN       $R;
        SLIT(3)    EPS;
        LOAD(3)    $C3;                % ACAR3:=EPS
        MLRN       $C3;
        LDS        $A;                 %RGS:=EPS*(ABS(H[L-1,L-1])
        LDL(3)     .H;
```

```
        LDA       *1(3);               % LDA  *H+1
        SAP       ;                    %RGA:=ABS(H[L,L-1])
        RTL       $A,1;
        LDA       $R;
        SBRN      $S;                  %I:=1  IF RGA IS NEGATIVE
        ISN       ;
        LIT(1)    =-1,1,0;
        LDL(3)    .MASK1;
        CAND(3)   $C0;                 % ACAR3:=0,0,N
        COR(1)    $C3;                 %ACAR1:=-1,1,N
        SETC(2)   I;                   % ACAR2:=I
BTO :   CTSBT(2)  0(1),CONT1;          % SKIP IF N-TH BIT OF $C2 IS 1
        TXGTM(1)  .BTO;
        CLC(1)    ;                    %$C1:=0 IF NO SUBDIAG ELE IS
%                                        SMALL
CONT1:  LDL(3)    .MASK1;
        CAND(1)   $C3;                 % $C1:=L
        CAND(3)   $C0;                 % $C3:=N
        CSUB(3)   $C1;                 %$C3:=N-L
        ZERT(3)   .ONEW;               % IF L=N, GO TO ONEW
        CSUB(3)   .ONE;
        ZERT(3)   .TWOW;               % IF L=N-1, GO TO TWOW
        SKIP      .AFTJ;
ONEW :  LDL(1)    .IT;        % ONE ROOT IS FOUND
        ZERT(1)   .ONEW1;              % IF IT=0, NO ITERATION COUNT
        LDL(2)    .ITS;                %       IS RECORDED
        LDL(3)    .MASK1;
        CAND(2)   $C3;                 % $C2:=ITERATION COUNT
        SETE      E OR -E;
        LDX       PEN;
        IXE       =0(0);
        SETE      I AND E;
        SETE1     E OR E;
        LDA       $C2;
        STA       0(1);           %IF IT IS NOT 0, IT:=ITERATION COUNT
ONEW1:  LIT(3)    0,0,-1;
        CADD(0)   $C3;                 % DECREASE ACARO BY 1
        JUMP      NEXTW;               %            AND GO TO NEXTW
TWOW :  LDL(1)    .IT;        % TWO ROOTS ARE FOUND
        ZERT(1)   .TWOW1;              % IF IT=0, NO ITERATION COUNT
        LDL(2)    .ITS;                %       IS RECORDED
        LDL(3)    .MASK1;
        CAND(2)   $C3;                 % $C2:=ITERATION COUNT
        SETE      E OR -E;
        LDX       PEN;
        IXE       =0(0);
        SETE      I AND E;
        SETE1     E OR E;
        LDA       $C2;
        STA       0(1);           %IF IT IS NOT 0, IT:=ITERATION COUNT
TWOW1:  LDL(2)    .SOLV;
        CSB(2)    0(0);
        STL(2)    .SOLV;               % N-TH BIT OF .SOLV:=1
        LIT(3)    0,0,-2;
        CADD(0)   $C3;                 % DECREASE ACARO BY 2
```

```
       JUMP       NEXTW;                  %        AND GO TO NEXTW
AFTJ : CAND(1)    .MASK1;                 % .L:=L
       STL(1)     .L;
       LDL(2)     .ITS;
       TXLF(2)    $C2,FAILJ;              % IF ITS=30 GO TO FAIL
       CAND(2)    .MASK1;
       LIT(1)     =10;
       CSUB(2)    $C1;
       ZERT(2)    .IT10;                  % IF ITS=10, GO TO IT10
       CSUB(2)    $C1;
       ZERT(2)    .IT10;                  % IF ITS=20, GO TO IT10
       SKIP       .ITN10;             % IF ITS NEQ 10,20,30, GO TO ITN10
FAILJ: JUMP       FAIL;
%  FORM SHIFT  S AND Y WHEN IT=10,20
IT10 : LDL(3)     .H;
       CADD(3)    $C0;
       LDA        0(3);                   % LDA H(0)
       SAP        ;
       LDS        $A;
       LDA        -1(3);                  % LDA  H-1(0)
       SAP        ;
       RTL        $A,1;
       LDA        $R;
       ADRN       $S;
       LDS        $A;                     %RGS:=ABS(H[N,N-1])
%                                                   +ABS(H[N-1,N-2])
       LIT(3)     =1.5;
       MLRN       $C3;
       IXE        =-1(0);
       SETE       I AND E;                %E:=1 IF RGX=N-1
       SETE       E OR E;
       LDC(2)     $A;
       STL(2)     .S;                     %.S:=1.5*RGS
       LDA        $S;
       MLRN       $S;
       LDC(3)     $A;
       STL(3)     .Y;                     %.Y:=RGS**2
       SKIP       .CONT15;
%  FORM SHIFT S AND Y WHEN IT NEQ 10,20
ITN10: LDL(3)     .H;
       CADD(3)    $C0;
       LDA        0(3);                   % LDA  H(0)
       LDS        -1(3);                  % LDS  H-1(0)
       RTL        $S,1;
       IXE        =0(0);
       SETG       I AND E;                %G:=1 FOR RGX=N
       SETE       I AND E;                %E:=G
       SETE1      E OR E;
       ADRN       $R;
       LDC(2)     $A;
       STL(2)     .S;                     %.S:=H[N,N]+H[N-1,N-1]
       SETE       E OR -E;
       SETE       H AND E;                %E:=1 IF RGX LEQ N
       SETE1      E OR E;
```

```
        LDL(3)    .H;
        CADD(3)   $C0;
        LDA       0(3);               % LDA  H(0)
        MLRN      $R;
        LDS       $A;
        LDA       0(3);               % LDA  H(0)
        RTL       $A,1;
        LDA       $R;
        SETE      G AND E;
        SETE1     E OR E;
        MLRN      -1(3);              % MLRN H-1(0)
        CHSA      ;
        ADRN      $S;
        LDC(3)    $A;
        STL(3)    .Y;                 %.Y:=H[N,N]*H[N-1,N-1]
%                                              -H[N,N-1]*H[N-1,N]
%  LOOK FOR TWO CONSECUTIVE SMALL DIAGONAL ELEMENT FOR M=N-2 TO L
CONT15:LDL(1)     .ITS;
        LDL(2)    .L;
        ALIT(1)   =1;                 % INCREASE ITERATION COUNT BY 1
        STL(1)    .ITS;               % AND STORE IN .ITS
        JXG       =0(2);
        IXL       =-1(0);
        SETE      E OR -E;
        SETE      J AND E;
        JXE       =0(2);
        SETE      J OR E;
        SETG      I AND E;            %G:=1 IF L LEQ RGX LEQ N-2
%  COMPUTE P=H[M,M]*(H[M,M]-S)+Y+H[M+1,M]*H[M,M+1]
        SETE      E OR -E;
        SETE      H AND E;            %E:=1 IF RGX LEQ N
        SETE1     E OR E;
        LDL(3)    .H;
        LDA       *0(3);              % LDA  *H
        LDS       $A;
        LDL(3)    .S;
        SBRN      $C3;
        SETE      G AND E;
        SETE1     E OR E;
        MLRN      $S;
        LDS       $A;
        SETE      E OR -E;
        SETE      H AND E;
        SETE1     E OR E;
        LDL(3)    .H;
        LDA       *-1(3);             % LDA  *H-1
        RTL       $A,-1;
        LDA       $R;
        SETE      G AND E;            %E:=1 IF L LEQ RGX LEQ N-2
        SETE1     E OR E;
        MLRN      *1(3);              % MLRN *H+1
        ADRN      $S;
        LDL(2)    .Y;
        ADRN      $C2;
        STA       P;
```

```
%  COMPUTE Q=H[M+1,M]*(H[M,M]+H[M+1,M+1]-S)
       SETE      E OR -E;
       SETE      H AND E;               %E;=1 FOR RGX LEQ N
       SETE1     E OR E;
       LDL(3)    .H;
       LDA       *0(3);                 % LDA  *H
       RTL       $A,-1;
       SETE      G AND E;               %E;=1 FOR L LEQ RGX LEQ N-2
       SETE1     E OR E;
       ADRN      $R;
       LDL(2)    .S;
       SBRN      $C2;
       MLRN      *1(3);                 % MLRN  *H+1
       STA       Q;
%  COMPUTE R=H[M+1,M]*H[M+2,M+1]
       SETE      E OR -E;
       SETE      H AND E;               %E;=1 FOR RGX LEQ N
       SETE1     E AND E;
       LDL(3)    .H;
       LDA       *1(3);                 % LDA  *H+1
       RTL       $A,-1;
       SETE      G AND E;               %E;=1 FOR L LEQ RGX LEQ N-2
       SETE1     E OR E;
       MLRN      $R;
       STA       R;
%  COMPUTE  P,Q,R=P,Q,R/(ABS(P)+ABS(Q)+ABS(R))
       LDA       P;
       SAP       ;                      %RGA;=ABS(P)
       LDS       $A;
       LDA       Q;
       SAP       ;                      %RGA;=ABS(Q)
       ADRN      $S;
       LDS       $A;
       LDA       R;
       SAP       ;
       ADRN      $S;
       LDS       $A;                    %RGS;=ABS(P)+ABS(Q)+ABS(R)
       LIT(2)    =1.0;
       CLRA      ;
       LDB       $A;
       LDA       $C2;                   %RGA;=1.0, RGB;=0
       DVRN      $S;
       LDS       $A;                    %RGS;=1.0/(ABS(P)+ABS(Q)+ABS(R))
       LDA       P;
       MLRN      $S;
       STA       P;                     %P;=P*RGS
       LDA       Q;
       MLRN      $S;
       STA       Q;                     %Q;=Q*RGS
       LDA       R;
       MLRN      $S;
       STA       R;                     %R;=R*RGS
%  TEST IF  ABS(H[M,M-1])*(ABS(Q)+ABS(R))
%              <EPS*(ABS(P)*ABS(H[M-1,M-1])+ABS(X)+ABS(Z))
```

```
SETE       E OR -E;
SETE       H AND E;
SETE1      E OR E;              %E:=1 FOR RGX LEQ N
LDA        Q;
SAP        ;
LDS        $A;
LDA        R;
SAP        ;
ADRN       $S;
LDS        $A;                  %RGS:=ABS(Q)+ABS(R)
LDL(3)     .H;
LDA        *1(3);               % LDA  *H+1
SAP        ;
RTL        $A,1;
LDL(2)     .L;
JXE        =0(2);               %J:=1 FOR RGX=L
 SETE      G AND E;
SETE       -J AND E;            %E:=1 FOR L<RGX LEQ N-2
SETI       E OR E;              %I:=1 FOR L<RGX LEQ N-2
 SETE1     E OR E;
LDA        $R;
MLRN       $S;
LDL(3)     .WR;
STA        0(3);                %WR:=(ABS(Q)+ABS(R))
SETE       E OR -E;
SETE       H AND E;             %E:=1 FOR RGX LEQ N
SETE1      E OR E;
LDL(3)     .H;
LDA        *0(3);               % LDA  *H
SAP        ;
LDS        $A;
RTL        $S,1;
SETE       I AND E;             %E:=1 FOR L<RGX LEQ N-2
SETE1      E OR E;
ADRN       $R;
RTL        $S,-1;
ADRN       $R;
LDS        $A;                  %RGS:=ABS(H[M+1,M+1])
LDA        P;                   %     +ABS(H[M,M])+ABS(H[M-1,M-1])
SAP        ;
SLIT(2)    EPS;
LOAD(2)    $C2;
MLRN       $C2;
MLRN       $S;                  %RGA:=EPS*ABS(P)*RGS
CHSA       ;
LDL(3)     .WR;
ADRN       0(3);                % ADRN  WR;
JSN        ;                    %J:=1 IF RGA IS NEGATIVE
LIT(1)     0,0,-2;
CADD(1)    $C0;
CAND(1)    .MASK1;              %ACAR1:=0,0,N-2
LIT(3)     =-1,0,0;
COR(1)     $C3;                 % ACAR1:=-1,0,N-2
LDL(3)     .L;
CADD(3)    .ONE;
```

```
        CSHL(3)    24;                       % ACAR3:=0,L+1,0
        COR(1)     $C3;                      % ACAR1:=-1,L+1,N-2
        SETC(3)    J;                        %ACAR3:=1 IF J CONTAINS 1
BT1   : CTSBT(3)   0(1),CONT2;
        TXGTM(1)   ,BT1;
        LDL(1)     ,L;
CONT2:  CAND(1)    ,MASK1;                   % ACAR1:=0,0,M
        STL(1)     ,M;                       %,M:=0,0,M
%   PUT ZERO IN APPROPRIATE PLACES
        SETE       E OR -E;
        SETE1      E OR E;
        CLRA       ;
        IXG       =0(1);
        JXE       =0(1);
        SETE       I AND E;
        SETE       J OR E;
        JXL       =-2(0);
        SETE       J AND E;                  % E,E1:=1 FOR M LEQ RGX < N-2
        SETE1      E OR E;
        LDL(3)     ,H;
        STA        +3(3);                    % STA   +H+3
        IXE       =-2(0);
        SETE       I OR E;                   % E,E1:=1 FOR M LEQ RGX LEQ N-2
        SETE1      E OR E;
        STA        +2(3);                    % STA   +H+2
%   PUT P,Q,R INTO ADB WHEN K=M
        SETE       E OR -E;                  %PICK UP M-TH ELEMENT OF
        SETE1      E OR E;                   % P,Q AND R WHEN K=M
        CLRA       ;
        IXE       =0(1);                     %I:=1 IF RGX=M
        SETE       I AND E;                  % E,E1:= 1
        SETE1      E OR E;
        LDA        P;
        LDS        Q;
        LDC(2)     $A;
        LDC(3)     $S;
        STL(2)     ,P;                       %,P:=M-TH ELEMENT OF P
        STL(3)     ,Q;                       %,Q:=M-TH ELEMENT OF Q
        LDA        R;
        LDC(2)     $A;
        STL(2)     ,R;
%   ARRANGE MATRIX INTO SKEW STORAGE FORM
        LIT(2)     -1,1,0;
        COR(2)     $C0;                      %ACAR2:=-1,1,N
        SETE       E OR -E;
        SETE1      E OR E;
DTAR1:  LDL(3)     ,H;
        CADD(3)    $C2;
        LDA        0(3);                     % LDA   H(2)
        RTL        $A,0(2);
        STR        0(3);                     % STR H(2)
        TXGTM(2)   ,DTAR1;
%   DOUBLE QR STEP INVOLVING ROWS L TO N AND COLUMNS M TO N
        LIT(3)    =0,1,-1;                   %INITIALIZATION OF K-LOOP
```

```
        CADD(3)    $C0;                    %ACAR3:=0,1,N-1
        CSHL(3)    24;                     %ACAR3:=1,N-1,0
        COR(1)     $C3;                    %ACAR1:=1,N-1,M
%  ACAR1 CONTROLS K-LOOP, K:M STEP 1 UNTIL N-1
NEXTK::LOL(2)      $C0;                    % SET NOTLAST WITH APPROPRIATE
        CSUB(2)    ,ONE;                   % BOOLEAN EXPRESSION
        CSUB(2)    $C1;                    %ACAR2:=(N-1)-K
        ZERT(2)    ,LAST;
        LOL(3)     ,ONE;
        STL(3)     ,NOTLAST;               % NOTLAST:=1 IF K NEQ N-1
        SKIP       ,CONT17;
LAST:   CLC(3)     ;
        STL(3)     ,NOTLAST;               %NOTLAST:=0 IF K=N-1
CONT17:
        SETE       E OR -E;
        SETE1      E OR E;
        CLRA       ;
        LOL(3)     ,MASK1;                 % TEST IF K=M
        CAND(3)    $C1;                    % ACAR3:=0,0,K
        CSUB(3)    ,M;                     % ,ACAR3:=K-M
        ZERT(3)    ,KEQM;                  % IF K=M GO TO KEQM,
                                           % P,Q,R ARE ALREADY COMPUTED
%  COMPUTE P=H[K,K-1],Q=H[K+1,K-1] AND R=H[K+2,K-1] WHEN K NEQ N
        LDX        PEN;
        RTL        $X,-1(1);
        LDX        $R;
        IXE        =2(1);
        SETH       I AND E;                %H:=1 FOR H[K+2,K-1]
        JXE        =1(1);                  %J:=1 FOR H[K+1,K-1]
        IXE        =0(1);                  %I:=1 FOR H[K,K-1]
        SETE       I AND E;
        SETE       J OR E;                 %G:=1 FOR H[K,K-1],H[K+1,K-1]
        SETG       H OR E;                 %                AND H[K+2,K-1]
        SETE       E OR -E;
        SETE       G AND E;
        SETE1      E OR E;                 % E,E1:=G
        LOL(3)     ,H;
        LOA        *0(3);                  % LDA   *H
        LOL(2)     ,NOTLAST;
        ZERF(2)    ,CPQR;
        SETE       H AND E;
        SETE1      E OR E;                 % IF NOTLAST=0, H[K+2,K-1]
        CLRA       ;                       % IS CONSIDERED TO BE 0
        SETE       E OR -E;
        SETE       G AND E;                % RESET E,E1
        SETE1      E OR E;
CPQR:   LOS        $A;                     %RGS:=H[K,K-1],H[K+1,K-1]
                                           %                   AND H[K+2,K-1]
        SAP        ;                       % RGA:=ABS(RGA)
        RTL        $A,1;
        ADRN       $R;
        RTL        $A,2;                  %RGA:=ABS(H[K,K-1])+ABS(H[K+1,K-1]
        ADRN       $R;                     %    +ABS(H[K+2,K-1]) AT H=1
        SETE       H AND E;                %E:=H
        SETE1      E OR E;
```

```
        LDC(2)     $A;                  %ACAR2:=RGA AT M=1
        STL(2)     .XX;                 % .XX:=ACAR2
        ZERF(2)    .NXPQ;
        JUMP       CONT3;               % IF .XX=0 GO TO CONT3
NXPQ:   SETE       E OR -E;
        SETE       G AND E;             % E,E1:=G
        SETE1      E OR E;
        CLRA       ;
        LDB        $A;
        LDA        $S;
        DVRN       $C2;
        LDS        $A;                  %RGS:=P/XX,Q/XX,R/XX
        SKIP       .CONT27;
%  PLACE P,Q,R IN APPROPRIATE REGISTER WHEN K=M
KEQM:   LDX        PEN;
        LDL(2)     .MASK1;
        CAND(2)    $C1;
        ZERT(2)    .KZERO;
        RTL        $X,-1(1);
        SKIP       .KEQM2;
KZERO:  RTL        $X,0(1);             % DUMMY BIT SETTING FOR COMPUTING
KEQM2:  LDX        $R;                  %                    S IN CONT27
        IXE        =2(1);
        SETH       I AND E;             %H:=1 FOR H[K+2,K]
        JXE        =1(1);               %J:=1 FOR H[K+1,K]
        IXE        =0(1);               %I:=1 FOR H[K,K]
        SETE       I AND E;
        SETE       J OR E;
        SETG       H OR E;          % G:=1 FOR H[K,K],H[K+1,K],H[K+2,K]
        SETE       E OR -E;
        LDL(2)     .P;
        LDL(3)     .Q;
        SETE       I AND E;
        SETE1      E OR E;              % E,E1:=I
        ADRN       $C2;                %RGA:=0,...,0,P,0,...,0
        SETE       E OR -E;
        SETE       J AND E;
        SETE1      E OR E;              %E,E1:=J
        ADRN       $C3;                %RGA:=0,...,0,P,Q,0,...,0
        LDL(2)     .R;
        SETE       E OR -E;
        SETE       H AND E;
        SETE1      E OR E;              %E,E1:=H
        ADRN       $C2;                %RGS,RGA:=0,...,0,P,Q,R,0,...,0
        SETE       E OR -E;
        SETE1      E OR E;
        LDS        $A;
%  COMPUTE S=SQRT(P**2+Q**2+R**2)
CONT27:
        SETC(2)    J;
        STL(2)     .JBIT;               %SAVE J BIT INTO .JBIT
        JSN        ;                    %J:=SIGN BIT OF RGA
        SETE       I AND E;
        SETJ       J AND E;             %J:=1 IF P IS NEGATIVE
```

```
      SETE      E OR -E;
      SETE      G AND E;               % E,E1:=G
      SETE1     E OR E;
      MLRN      $A;               %$A:=(P/XX)**2,(Q/XX)**2,(R/XX)**2
      RTL       $A,1;
      ADRN      $R;
      RTL       $A,2;
      ADRN      $R;               %$A:=(P/XX)**2+(Q/XX)**2+(R/XX)**2
      SETE      E OR -F;
      SETE      H AND E;
      SETE1     E OR E;
      STL(0)    .SV0;                  % SAVE ACAR0 INTO HQSV0
      STL(1)    .SV1;                  % SAVE ACAR1 INTO HQSV1
      SETC(0)   I;                     % SAVE I-BIT
      SETC(1)   J;                     % SAVE J-BIT
      STL(0)    .SAVI;
      STL(1)    .SAVJ;
      CLC(3)    ;
      SLIT(3)   SQRT;                  % CALL SQRT SUBROUTINE
      EXCHL(3)  $ICR;                  %$A:=SQRT($A)
      LDL(0)    .SAVI;
      LDL(1)    .SAVJ;
      LDI       $C0;
      LDJ       $C1;
      LDL(0)    .SV0;
      LDL(1)    .SV1;                  % RESTORE ACAR0 AND ACAR0
      SETC(2)   J;                     % ACAR2:=J BIT
      ZERT(2)   .PPOS;                 %SKIP TO PPOS IF ACAR2 HAS ALL 0
      CHSA      ;                     %RGA:=-RGA IF J BIT HAS 1
PPOS : LDC(2)   $A;
      STL(2)    .S;                    %.S:=SQRT((P/XX)**2+(Q/XX)**2
                                       %                    +(R/XX)**2)
      LDL(3)    .JBIT;
      LDJ       $C3;                   % RESTORE J-BIT WITH .JBIT
%  COMPUTE X,Y,Z FOR ROW AND COLUMN MODIFICATION
      SETE      E OR -E;
      SETE      I AND E;               % E,E1:=I
      SETE1     E OR E;
      LDA       $S;
      ADRN      $C2;                   %$A:=P+.S WHERE I=1
      LDS       $A;
      LDC(3)    $A;
      STL(3)    .P;                    %.P:=.P+.S
      SETE      E OR -F;
      SETE      G AND F;
      SETE1     E OR E;                % E,E1:=G
      CLRA      ;
      LDB       $A;
      LDA       $S;
      DVRN      $C2;                   %RGA:=.P/.S,.Q/.S,.R/.S
      SETE      I AND E;               % E,E1:=I
      SETE1     E OR E;
      LDC(3)    $A;
      STL(3)    .X;                    %.X:=.P/.S
      SETE      E OR -E;
```

```
        SETE      J AND E;             % E.E1:=J
        SETE1     E OR E;
        LDC(2)    $A;
        STL(2)    .Y;                  %.Y:=.Q/.S
        SETE      E OR -E;
        SETE      H AND E;             % E.E1:=H
        SETE1     E OR E;
        LDC(3)    $A;
        STL(3)    .Z;                  %.Z:=.R/.S
%  COMPUTE Q=Q/P AND R=R/P FOR ROW AND COLUMN MODIFICATION
        SETE      E OR -E;
        SETE      J AND E;
        SETE      H OR E;              % E.E1:=J OR H
        SETE1     E OR E;
        LDL(3)    .P;
        CLRA      ;
        LDB       $A;
        LDA       $S;
        DVRN      $C3;                 %RGA:=.Q/.P..R/.P FOR J=1 AND H=1
        SETE      J AND E;
        SETE1     E OR E;
        LDC(2)    $A;
        STL(2)    .Q;                  %.Q:=.Q/.P
        SETE      E OR -E;
        SETE      H AND E;
        SETE1     E OR E;
        LDC(3)    $A;
        STL(3)    .R;                  %.R:=.R/.P
%  COMPUTE NEW H[K.K-1]
%   IF K NEQ M THEN H[K,K-1]:= -S*X
%          ELSE IF L NEQ M THEN H[K.K-1]:= -H[K.K-1]
        LDL(2)    .MASK1;
        CAND(2)   $C1;                 %ACAR2:=0.0.K
        CSUB(2)   .M;                  %ACAR2:=0.0.K-M
        ZERT(2)   .CPLM;               % IF K=M SKIP TO COMPARISON
        SETE      E OR -E;             %   BETWEEN L AND M
        SETE      I AND E;
        SETE1     E OR E;              % E.E1:=I
        CLRA      ;                 %IF K NEQ M. H[K.K-1]:=-S*X
        LDL(2)    .S;
        LDL(3)    .XX;
        SBRN      $C2;
        MLRN      $C3;
        LDL(3)    .H;
        STA       *0(3);               % STA  *H
        SKIP      .RWMOD;
CPLM :  LDL(2)    .L;
        CSUB(2)   .M;                  % TEST IF L=M
        ZERT(2)   .RWMOD;          %H[K.K-1]:=-H[K.K-1] IF L NEQ M.K=M
        LDX       PEN;
        RTL       $X.-1(1);            % ACAR1 CONTAINS M WHICH IS
        LDX       $R;                  %         GREATER THAN L
        IXE       =0(1);
        SETE      E OR -E;
```

```
      SETE      I AND E;
      SETE1     E OR E;
      LDL(3)    .H;
      LDA       *0(3);              % LDA  *H
      CHSA      ;
      STA       *0(3);              % STA  *H
%
   DEFINE  KRGXN =
      IXG       =0(1);              %I:=1 IF K<RGX
      JXL       =0(0);              %J:=1 IF RGX<N
      SETE      I AND E;
      SETE      J AND E;            %E:=1 IF K<RGX<N
      IXE       =0(1);              %I:=1 IF RGX=K
      JXE       =0(0);              %J:=1 IF RGX=N
      SETE      I OR E;          % E.E1 ARE SET WHEN K LEQ RGX LEQ N
      SETE      J OR E ##;
%
RWMOD: SETE     E OR -E;
%  ROW MODIFICATION
      LDX       PEN;
      RTL       $X.2(1);
      LDX       $R;
      KRGXN     ;
      SETH      E OR E;             % H IS SET FOR H[K+2.J]
      SETE      E OR -E;
      RTL       $X,-1;
      LDX       $R;
      KRGXN     ;
      SETG      E OR E ;            % G IS SET FOR H[K+1.J]
      RTL       $X,-1;
      SETE      E OR -E;
      LDX       $R;
      KRGXN     ;
      SETJ      E OR E;             % J IS SET FOR H[K.J]
      SETE      E OR -E;
      SETE1     E OR E;
      LDL(3)    .H;
      CADD(3)   $C1;
      LDA       1(3);               % LDA H+1(1),  RGA:=H+1(1)
      SETE      G AND E;            % E,E1:=G
      SETE1     E OR E;
      LDL(2)    .Q;
      MLRN      $C2;
      LDS       $A;                 % RGS:=.Q*RGA
      SETE      E OR -E;
      SETE1     E OR E;             % E,E1:=1
      LDA       0(3);               % LDA  H(1)
      RTL       $A,1;               % ROUTE RGA BY 1
      LDA       $R;
      ADRN      $S;
      LDS       $A;                 % RGS:=RGS+RGR
      LDL(2)    .NOTLAST;
      ZERT(2)   .CONT4;             % IF NOTLAST=0 GO TO CONT4
%  IF NOTLAST=1  COMPUTE
%     H[K+2.J]:= H[K+2,J]-(H[K,J]+Q*H[K+1.J]+R*H[K+2,J])*Z
```

```
         LDA       2(3);              %RGA:=H+2(1)
         RTL       $S,1;              % ROUTE RGS BY 1
         LDS       $R;
         SETE      H AND E;
         SETE1     E OR E;            % E,E1:=H
         LDL(2)    .R;
         LDL(3)    .Z;
         MLRN      $C2;
         ADRN      $S;
         LDS       $A;                % RGS:=RGA*.R+RGS
         MLRN      $C3;               % RGA:=.Z*RGS
         CHSA      ;
         LDL(3)    .H;
         CADD(3)   $C1;
         ADRN      2(3);              % ADRN  H+2(1)
         STA       2(3);              %H+2(1):=H+2(1)-RGA
         SETE      E OR -E;
         SETE1     E OR E;
         RTL       $S,-1;
         LDS       $R;                % ROUTE RGS BY -1
%  COMPUTE
%   H[K+1,J]:= H[K+1,J]-(H[K,J]+Q*H[K+1,J]+R*H[K+2,J])*Y
CONT4: SETE        E OR -E;
         SETE      G AND E;
         SETE1     E OR E;            % E,E1:=G
         LDL(2)    .Y;
         LDA       $S;
         CHSA      ;
         MLRN      $C2;               % RGA:=-RGS*.Y
         LDL(3)    ,H;
         CADD(3)   $C1;
         ADRN      1(3);              % ADRN  H+1(1)
         STA       1(3);              % STA   H+1(1)
%  COMPUTE
%    H[K,J]:= H[K,J]-(H[K,J]+Q*H[K+1,J]+R*H[K+2,J])*X
         SETE      E OR -E;
         SETE      J AND E;
         SETE1     E OR E;            % E,E1:=J
         RTL       $S,-1;             % ROUTE RGS BY -1
         LDL(3)    ,X;
         LDA       $R;
         CHSA      ;
         MLRN      $C3;               %RGA:=-RGS*,X
         LDL(3)    ,H;
         CADD(3)   $C1;
         ADRN      0(3);              % ADRN  H(1)
         STA       0(3);              %H(1):=H(1)+RGA
%
%  COLUMN MODIFICATION
         LDL(2)    ,MASK1;            % SET J WITH K+3 OR N
         LIT(3)    0,0,3;
         CAND(2)   $C1;               % ACAR2:=0,0,K
         CADD(2)   $C3;               % ACAR2:=0,0,K+3
         LDL(3)    $C2;               % ACAR3:=0,0,K+3
```

```
       CSUB(2)    $C0;                 % ACAR2:=0.0.K+3-N
       CTSBF(2)   40.NINTOUJ;          % IF 40-TH BIT OF ACAR2 IS 0.J:=N
       STL(3)     .J;                  %.J:=K+3 IF K+3<N
       SKIP       .CONT5;              %.J:=N IF K+3 GEQ N
NINTOUJ:STL(0)    .J;
%
    DEFINE  LRGXJ =
       LDL(2)     .L;
       LDL(3)     .J;
       IXG        =0(2);               %I:=1 IF L<RGX
       JXL        =0(3);               %J:=1 IF RGX<J
       SETE       E OR -E;
       SETE       I AND E;
       SETE       J AND E;             %E:=1 IF L<RGX<J
       IXE        =0(2);               %I:=1 IF L=RGX
       JXE        =0(3);               %J:=1 IF RGX=J
       SETE       I OR E;
       SETE       J OR E ##;           % E:=1 IF L LEQ RGX LEQ J
%
CONT5: SETE       E OR -F;
       LDX        PEN;                 %RGX:=PEN
       RTL        $X.0(1);             % ROUTE RGX BY K
       LDX        $R;
       LRGXJ      ;
       SETH       E OR E;              % H IS SET FOR H[I.K]
       SETE1      E OR E;
       LDL(2)     .X;
       LDL(3)     .H;
       LDA        *0(3);               %RGA:=H[I.K]
       MLRN       $C2;
       LDS        $A;                  % RGS:=H[I.K]*X
       SETE       E OR -F;
       SETE1      E OR E;
       RTL        $X.1;
       LDX        $R;                  % ROUTE RGX BY 1
       RTL        $S.1;
       LDS        $R;                  % ROUTE RGS BY 1
       LRGXJ      ;
       LDL(2)     .Y;
       SETG       E OR E;              % G IS SET FOR H[I.K+1]
       SETE1      E OR E;
       LDL(3)     .H;
       LDA        *0(3);               %RGA:=H[I.K+1]
       MLRN       $C2;                 % $A:=H[I.K+1]*Y
       ADRN       $S;
       LDS        $A;                  %$S:=X*H[I.K]+H[I.K+1]
%  COMPUTE H[I.K+2]=H[I.K+2]-(X*H[I.K]+Y*H[I.K+1]+Z*H[I.K+2])*R
%                                                   IF NOTLAST=1
       SETE       E OR -E;
       SETE1      E OR E;
       LDL(2)     .NOTLAST;            % IF NOTLAST=0. GO TO CONT6
       ZERT(2)    .CONT6;              % NO NEED TO COMPUTE H[I.K+2]
       RTL        $X.1;
       LDX        $R;
       RTL        $S.1;
```

```
       LDS        $R;
       LRGXJ      ;
       SETE1      E OR E;                 %E,E1:=1 FOR H[I,K+2]
       LDL(2)     ,Z;
       LDL(3)     ,H;
       LDA        *0(3);                  %RGA:=H[I,K+2]
       LDL(3)     ,R;
       MLRN       $C2;                    %RGA:=Z*H[I,K+2]
       ADRN       $S;                     %RGS:=X*H[I,K]+Y*H[I,K+1]
       LDS        $A;                     %                  +Z*H[I,K+2]
       CHSA       ;
       MLRN       $C3;
       LDL(3)     ,H;
       ADRN       *0(3);                  % ADRN   *H
       STA        *0(3);                  % STA    *H
       SETE       E OR -E;
       SETE1      E OR E;
       RTL        $X,-1;
       LDX        $R;
       RTL        $S,-1;
       LDS        $R;
%  COMPUTE H[I,K+1]=H[I,K+1]-(X*H[I,K]+Y*H[I,K+1]+Z*H[I,K+2])*R
CONT6: SETE       G AND E;
       SETE1      E OR E;                 % E,E1:=G
       LDL(2)     ,Q;
       LDA        $S;
       CHSA       ;
       MLRN       $C2;
       LDL(3)     ,H;
       ADRN       *0(3);                  % ADRN   *H
       STA        *0(3);                  % STA    *H
%  COMPUTE H[I,K]:=[I,K]-(X*H[I,K]+Y*H[I,K+1]+Z*H[I,K+2])
       SETE       E OR -E;
       SETE1      E OR E;
       RTL        $X,-1;
       LDX        $R;                     % ROUTE RGX BY -1
       RTL        $S,-1;                  % ROUTE RGS BY -1
       LDA        $R;
       SETE       H AND E;                % E,E1:=H
       SETE1      E OR E;
       CHSA       ;
       LDL(3)     ,H;
        ADRN       *0(3);                      % ADRN   *H
       STA        *0(3);                  % STA   *H,RGA:=H[I,K]-RGS
CONT3::TXLTM(1) ,NXKJ;
       SKIP       ,NXNJ;
NXKJ : JUMP       NEXTK;
%  END OF K-LOOP
%  REARRANGE MATRIX INTO STANDARD FORM
NXNJ : LIT(1)     -1,1,0;
       CADD(1)    $C0;                  %$C1:=-1,0,N
       SETE       E OR -E;
       SETE1      E OR E;
DTAR2: LDL(3)     ,H;
```

```
       CADD(3)    $C1;
       LDA        0(3);                 % LDA   H(1)
       LIT(2)     =0.0.64;
       CSUB(2)    $C1;
       RTL        $A.0(2);
       STR        0(3);                 % STR   H(1)
       TXGTM(1)   .OTAR2;               % TEST AND MODIFY ACAR2 BY -1
       JUMP       NEXTIT;
% IF ALGORITHM FAILS.G:=1 WHERE SUBDIAGONAL ELEMENTS CONVERGED
FAIL ::SETE       E OR -E;
       LDX        PEN;
       IXG        =0(0);
       LDL(3)     .N;
       JXL        =0(3);                % JXL   =N
       SETE       I AND E;
       SETE       J AND E;
       JXE        =0(3);                % JXE   =N
       SETG       J OR E;               % ACARO CONTAINS PRESENT MATRIX
       JUMP       EIG;                  % SIZE
%  IF PROCESS IS COMPLETED.G:=1 FOR ALL BITS
FIN  ::SETE       E OR -E;
       LDX        PEN;
       IXG        =0;
       LDL(3)     .N;
       JXL        =0(3);                % JXL   =N
       SETE       I AND E;
       SETE       J AND E;
       IXE        =0;
       JXE        =0(3);                % JXE   =N
       SETE       I OR E;
       SETG       J OR E;               %G:=1 FOR 0 LEQ RGX LEQ N
%  FIND  ONE ROOT
EIG  ::SETE       E OR -E;
       LDL(3)     .N;
       IXL        =0(3);                % IXL   =N
       JXE        =0(3);                % JXE   =N
       SETE       I AND E;
       SETE       J OR E;
       SETE       G AND E;
       SETE1      E OR E;
       LDL(3)     .H;
       LDA        *0(3);                % LDA   *H
       LDL(3)     .WR;
       STA        0(3);                 % STA   WR
       CLRA       ;
       LDL(3)     .WI;
       STA        0(3);                 % IMAGINARY PART:= 0
%  TWO ROOTS ARE FOUND
       LDL(2)     .SOLV;                % SET APPROPRIATE BIT PATTERN
       ZERF(2)    .TWRT;
       JUMP       HOREND;
TWRT : SETE       E OR -E;
       IXG        =0;
       LDL(3)     .N;
       JXL        =0(3);                % JXL   =N
```

```
      SETE       I AND E;
      SETE       J AND E;
      IXE        =0;
      JXE        =0(3);             % JXE  =N
      SETE1      J OR E;
      SETH       G AND E1;          %H:=1 IF 1 LEQ RGX LEQ N AND G=1
      SETE1      I OR E;
      SETJ       G AND E1;       %J:=1 IF 0 LEQ RGX LEQ N-1 AND G=1
      SETE       E OR -E;
      SETE       H AND E;
      SETG       J OR E;                % G:=1 IF 0 LEQ RGX LEQ N AND G=1
      SETE       E OR -E;
      SETE       J AND E;           % E:=J
      SETE1      E OR E;
      LDL(3)     .H;
      LDA        *1(3);             % LDA  *H+1
      RTL        $A,1;              % ROUTE RGA BY 1
      SETE       E OR -E;
      SETE       H AND E;
      SETE1      E OR E;            % E:=H
      LDA        *-1(3);            % LDA  *H-1
      MLRN       $R;
      STA        TMP1;              % TMP1:=H[N-1,N]*H[N,N-1]
      SETE       E OR -E;
      SETE       G AND E;           % E,E1:=G
      SETE1      E OR E;
      LDL(3)     .H;
      LDA        *0(3);             % LDA  *H
      RTL        $A,1;              % ROUTE RGA BY 1
      SETE       H AND E;
      SETE1      E OR E;            % E,E1:=H
      LDS        $R;
      ADRN       $R;
      SHAMR      1;
      NORM       ;
      STA        TMP2;              %TMP2:=(H[N,N]+H[N-1,N-1])/2
      LDA        $S;
      LDL(3)     .H;
      SBRN       *0(3);             % SBRN  *H
      SHAMR      1;
      NORM       ;
      MLRN       $A;                %RGA:=((H[N-1,N-1]-H[N,N])/2)**2
      ADRN       TMP1;              % RGA:=RGA+H[N-1,N-1]*H[N,N]
      ISN        ;                  %I:=SIGN OF RGA
      SETC(0)    I;                 % ACARO:= I BIT
      SAP        ;                  % RGA:= ABS(RGA)
      STL(0)     .SVO;              % SAVE ACARO
      CLC(3)     ;
      SLIT(3)    SQRT;
      EXCHL(3)   $ICR;              % RGA:=SQRT(ABS(RGA))
      STA        TMP3;              % TMP3:=RGA
      LDL(0)     .SVO;              % RESTORE ACARO
%  FIND COMPLEX PAIR
      SETC(1)    H;                 % ACAR1,ACAR2:=H BIT
```

```
        LDL(2)     $C1;
        CAND(1)    $C0;                 % ACAR1:=H BIT AND SIGN
        CAND(1)    .SOLV;               % ACAR1:=ACAR1 AND SLTN PATTERN
        ZERT(1)    .FREAL;
        LDFE1      $C1;
        LDL(3)     .WI;
        STA        0(3);                % STA  WI
        LDS        TMP2;                % FOUND WR[N] AND WI[N]
        LDL(3)     .WR;
        STS        0(3);                % STA  WR
        CSHL(1)    1;                   % SHIFT ACAR1 LEFT BY 1
        LDEE1      $C1;                 % E,E1:=ACAR1
        RTL        $A,-1;               % ROUTE RGA BY -1
        LDA        $R;
        RTL        $S,-1;               % ROUTE RGS BY -1
        CHSA       ;
        LDL(3)     .WI;
        STA        0(3);                % STA  WI
        LDL(3)     .WR;
        STR        0(3);                % STR  W
%  FIND REAL PAIR
FREAL;  COMPC(0)   ;                    % ACAR0:=COMPLEMENT OF ACAR0
        CAND(2)    $C0;                 % ACAR2:=H AND C(SIGN)
        CAND(2)    .SOLV;
        ZERT(2)    .HQREND;
        LDEE1      $C2;
        LDA        TMP2;
        LDS        $A;
        ADRN       TMP3;
        LDL(3)     .WR;
        STA        0(3);                % STA  WR
        CLRA       ;
        LDL(3)     .WI;
        STA        0(3);                % STA  WI
        LDA        TMP3;
        CHSA       ;
        ADRN       $S;                  % RGA:=TMP2-TMP3
        CSHL(2)    1;
        LDEE1      $C2;
        RTL        $A,-1;
        LDL(3)     .WR;
        STR        0(3);                % STR  WR
        CLRA       ;
        LDL(3)     .WI;
        STA        0(3);                % STA WI
HQREND:;
        LDL(0)     .HQSVE;
        LDEE1      $C0;                 % RESTORE E,E1
        LDL(0)     .HQSV0;
        LDL(1)     .HQSV1;
        LDL(3)     .HQSV3;
        EXCHL(3)   $ICR;
        END        HQR.
```

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

THE QR-ALGORITHM

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Research Report

5. AUTHOR(S) (First name, middle initial, last name)

Masako Ogura

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1, 1971	44	

8a. CONTRACT OR GRANT NO. USAF 30-(602)-4144	9a. ORIGINATOR'S REPORT NUMBER(S) CAC Document No. 12
b. PROJECT NO. ARPA Order 788	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Copies may be requested from the address given in (1) above.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Rome Air Development Center Griffiss Air Force Base Rome, New York 13440

13. ABSTRACT

The implementation of QR-algorithm on ILLIAC IV is described. An ASK subroutine for computing all eigenvalues of a real Hessenberg matrix of order less than or equal to 64 by this algorithm is attached. The QR-transformation consists of the decomposition of the matrix A_k into the product of a unitary matrix Q_k and an upper triangular matrix R_k, and forming A_{k+1} by post-multiplying R_k by Q_k, where $A_1 = A$ is the original matrix. All eigenvalues are either isolated on the diagonal or are eigenvalues of a 2 x 2 diagonal submatrix as $k \to \infty$.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Matrix Algebra						

UNCLASSIFIED

Security Classification